A Belief-Driven Method for Discovering Unexpected Patterns

Balaji Padmanabhan
Department of Information Systems
Leonard N. Stern School of Business
New York University
bpadmana@stern.nyu.edu

Alexander Tuzhilin[1]
Computer Science Department
Columbia University
tuzhilin@cs.columbia.edu

Abstract

Several pattern discovery methods proposed in the data mining literature have the drawbacks that they discover too many obvious or irrelevant patterns and that they do not leverage to a full extent valuable prior domain knowledge that decision makers have. In this paper we propose a new method of discovery that addresses these drawbacks. In particular we propose a new method of discovering unexpected patterns that takes into consideration prior background knowledge of decision makers. This prior knowledge constitutes a set of expectations or beliefs about the problem domain. Our proposed method of discovering unexpected patterns uses these beliefs to seed the search for patterns in data that contradict the beliefs. To evaluate the practicality of our approach, we applied our algorithm to consumer purchase data from a major market research company and to web logfile data tracked at an academic Web site and present our findings in the paper.

1. Introduction

The field of knowledge discovery in databases (data mining) has been defined in [FPS96] as the non-trivial process of identifying *valid*, *novel*, *potentially useful*, and *ultimately understandable* patterns from data. However, most of the work in the KDD field focuses on the *validity* aspect, and the other two aspects, *novelty* and *usefulness*, were studied to a lesser degree. This is unfortunate because it has been observed both by researchers [FPM91, KMR+94,BMU+97,ST95,ST96a,LH96] and practitioners [S97,F97] that many existing tools generate a large number of valid but *obvious* or *irrelevant* patterns. To address this issue, some researchers have studied the discovery of novel [ST95,ST96a,LH96,LHC97,PT97a] and useful [PSM94, ST95, ST96a,AT97] patterns.

In this paper, we continue the former stream of research and focus on the discovery of *unexpected* patterns. Unexpectedness of a rule relative to a belief system has been considered before in [ST95,ST96a,LH96,LHC97, PT97a]. In [ST95,ST96a] "unexpectedness" of a rule is defined relative to a system of user-defined beliefs. A rule is considered to be "interesting" if it affects the degrees of beliefs. Therefore, unexpectedness is defined in probabilistic terms in [ST95, ST96a]. Liu and Hsu take a different approach to defining unexpectedness in [LH96]. In particular, [LH96] captures a measure of rule "distance" and is based on a *syntactic* comparison between a rule and a belief. In [LH96], a rule and a belief are "different" if either the consequents of the rule and the belief are "similar" but the antecedents are "far apart" or vice versa, where "similarity" and "difference" are defined syntactically based on the structure of the rules. In addition, [LHC97] proposes a method in which users can specify their beliefs by using "generalized impressions" that are easier for the user to specify than specific beliefs. However the discovery method again is based on syntactic comparisons of rules and beliefs. This does not capture the concept of "unexpectedness" in terms of logical contradiction of rules and beliefs as argued in [PT97a] to be better.

In [PT97a] we proposed a new definition of unexpectedness in terms of a *logical contradiction* of a rule and a belief. In this paper, we take this approach and formally present an algorithm for discovering unexpected patterns. We also test this algorithm on data provided to us by a major market research company and on Web logfile data gathered at an academic website and present our findings. We also demonstrate that our method provides a simple, yet effective way to discovering interesting patterns in the data.

In this paper, we focus only on the discovery of unexpected patterns given an initial set of beliefs. We *do not* address the issue of how to build a "good" set of beliefs. We assume that it can be generated using methods described in [ST96b], such as elicitation of beliefs from the domain expert, learning them from data, and refinement of existing beliefs using newly discovered patterns. A similar issue of how to specify an initial set of beliefs has also been addressed in [LHC97].

2. Unexpectedness of a Rule

In order to define the concept of unexpectedness, we first present some preliminaries. We consider rules and beliefs of the form $X \rightarrow A$, where X and A are conjunctions of literals (i.e., either atomic formulas of first-order logic or

[1] On sabbatical leave from NYU.

negations of atomic formulas). We keep this definition general and do not impose restrictions of the structures of atomic formulas that can appear in literals of X and A. We also associate with the rule some measure of its statistical "strength" [PS91], such as "confidence" and "support" [AIS93]. We say that a rule *holds* on a dataset if the "strength" of the rule is greater than a user-defined threshold value.

We also make an assumption of *monotonicity of beliefs*. In particular, if we have a belief $Y \rightarrow B$ that we expect to hold on a dataset D, then the belief will also be expected to hold on any "statistically large"[2] subset of D. If we have a non-monotonic belief (that we expect *not* to hold for some subset of the data), we incorporate our knowledge of why we do not expect the belief to hold on the subset into the belief, thereby making the belief more specific (as shown in [PT97b]). We can do this iteratively until we have a set of monotonic beliefs.[3] Given these preliminary concepts, we define unexpectedness of a rule.

Definition. The rule $A \rightarrow B$ is *unexpected* with respect to the belief $X \rightarrow Y$ on the dataset D if the following conditions hold:

(a) B *AND* Y |= FALSE. This condition states that B and Y logically contradict each other.
(b) A *AND* X holds on a statistically large[2] subset of tuples in D. We use the term "*intersection of a rule with respect to a belief*" to refer to this subset. This intersection defines the subset of tuples in D in which the belief and the rule are both "applicable" in the sense that the antecedents of the belief and the rule are both true on all the tuples in this subset.
(c) The rule $A, X \rightarrow B$ holds. Since condition (a) constrains B and Y to logically contradict each other, it follows that the rule $A, X \rightarrow \neg Y$ holds. □

We believe that this definition captures the spirit of "unexpectedness" for the following reasons:
(1) The heads of the rule and the belief are such that they logically contradict each other. Therefore in *any* tuple where the belief and the rule are both "applicable," if the rule holds on this tuple, the belief cannot hold and vice-versa.
(2) Since both a rule and a belief hold *statistically*, it is inappropriate to label a rule "unexpected" if the intersection of the contradicting rule and the belief is very small. Hence we impose the condition that the intersection of the belief and the rule should be statistically large. Within this statistically large intersection, we would expect our belief to hold because of the *monotonicity* assumption. However if the rule holds in this intersection, the belief cannot hold because the heads of the rule and belief logically contradict each other. Hence the expectation that the belief should hold on this statistically large subset is contradicted. We next present an algorithm, which is an extension of standard association rule generating algorithms [AMS+95] for finding unexpected rules.

3. Discovery of Unexpected Rules

In this section we present an algorithm for discovering unexpected rules. The rules we consider are of the form *body* $\rightarrow$ *head*, where *body* is a conjunction of atomic conditions of the form *attribute op value* and *head* is a single atomic condition of the form *attribute op value*, where $op \in \{\geq, \leq, =\}$. This definition extends the structure of association rules [AMS+95] by considering discrete domains and conditions involving comparison operators $\geq$ and $\leq$. We consider these extensions since in many applications, such as the Web logfile application, rules and beliefs involve these additional operators. We further follow the approach taken in [AMS+95] and discover unexpected rules that satisfy user-specified minimum *support* and *confidence*[4] requirements.

We note that some discrete attributes in the domain may be *unordered* (e.g. "Country"). When an unordered attribute is part of a condition, we restrict the operator in that condition to be "=" (we disallow conditions such as "*country* > *Brazil*", since *country* is an unordered attribute).

3.1 Overview of the Discovery Strategy

Consider a belief $X \rightarrow Y$ and a rule $A \rightarrow B$, where both X and A are conjunctions of atomic conditions and both Y and B are single atomic conditions. It follows from the definition of unexpectedness in Section 2 that if a rule $A \rightarrow B$ is "unexpected" with respect to the belief $X \rightarrow Y$, then the rule $X, A \rightarrow B$ also holds. We propose the discovery algorithm *ZoomUR* ("Zoom to Unexpected Rules") that consists of two parts: *ZoominUR* and *ZoomoutUR*. Given a belief $X \rightarrow Y$, algorithm *ZoomUR* first discovers (in *ZoominUR*) all rules (satisfying threshold support and confidence requirements) of the form $X, A \rightarrow B$, such that B contradicts the head of the belief. We *then* consider (in *ZoomoutUR*) other more general and potentially unexpected rules of the form $X', A \rightarrow B$, where $X' \subset X$.

[2] In this paper, we use a user-specified *support* threshold value to determine if the subset is large enough.
[3] Converting non-monotonic beliefs to monotonic beliefs can be automated by letting the user specify non-monotonic beliefs with *exceptions*. Then the system automatically converts these to a set of monotonic beliefs.

[4] Rule *body* $\rightarrow$ *head* holds in a dataset with confidence c if c% of the transactions containing *body* also contain *head*; the rule has support s if s% of transactions contain *body* and *head* [AIS93].
[5] Since we consider attributes with a finite number of discrete levels in this paper, the number of conditions involving these levels are also finite.

The rules that *ZoominUR* discovers are "refinements" to the beliefs such that the beliefs are contradicted. The rules that *ZoomoutUR* discovers are *not* refinements, but more general rules that satisfy the conditions of unexpectedness. For example, if a belief is that "*professional→ weekend*" (professionals tend to shop more on weekends than on weekdays), *ZoominUR* may discover a refinement such as "*professional, december → weekday*" (in December, professionals shop more on weekdays than on weekends). *ZoomoutUR* may then discover a more general rule "*december→weekday*", totally different from the belief "*professional→ weekend*".

3.2 Algorithm ZoominUR

Algorithm ZoominUR is based on algorithm Apriori's ideas [AMS+95] of generating association rules from *itemsets* in an incremental manner. In this paper we use the term "itemset" to refer to a conjunction of atomic conditions, each of the form *attribute op value* where $op \in \{\geq, \leq, =\}$. An itemset is said to be *large* if the percentage of transactions that satisfy the conjunction of conditions exceeds the user-specified minimum support level. There are two main extensions to Apriori that we make in ZoominUR: (1) ZoominUR starts with a set of initial beliefs to seed the search for unexpected rules. This is similar in spirit to the work of [SVA97] where itemset constraints are used to focus the search. (2) We incorporate comparisons since in many applications some rules involve these operators. Before presenting ZoominUR, we first explain some preliminaries.

Consider the belief $X \rightarrow Y$, where X and Y are as defined in Section 3.1. We use the term *"CONTR(Y)"* to refer to the set of atomic conditions of the form *attribute op value* that contradict Y, where $op \in \{\geq, \leq, =\}$. Assume that the head of the belief is *a op val*, where a is an attribute in the domain. Further assume that $v_1, v_2,...,v_k$ are the set of unique discrete values (sorted in ascending order if a is ordered) that the attribute a takes on in D. *CONTR(Y)* is generated as follows:

(1) If the head of the belief is of the form "$a \geq val$":
- a) Any condition of the form "$a \leq v_p$"$\in$ *CONTR(Y)* if $v_p \in \{v_1,v_2,...v_k\}$ and $v_p < val$; (e.g. the head "month ≥ 10" is contradicted by "month ≤ x", where x could be from {1,2,...,9})
- a) Any condition of the form "$a = v_p$"$\in$ *CONTR(Y)* if $v_p \in \{v_1,v_2,...v_k\}$ and $v_p < val$;

(2) If the head of the belief is of the form "$a \leq val$":
- a) Any condition of the form "$a \geq v_p$"$\in$ *CONTR(Y)* if $v_p \in \{v_1,v_2,...v_k\}$ and $v_p > val$;
- a) Any condition of the form "$a = v_p$"$\in$ *CONTR(Y)* if $v_p \in \{v_1,v_2,...v_k\}$ and $v_p > val$;

(3) If the head of the belief is of the form "$a = val$":
- a) If a is an ordered attribute, "$a \geq v_p$"$\in$ *CONTR(Y)* if $v_p \in \{v_1,v_2,...v_k\}$ and $v_p > val$;
- a) If a is an ordered attribute, "$a \leq v_p$"$\in$ *CONTR(Y)* if $v_p \in \{v_1,v_2,...v_k\}$ and $v_p < val$;
- a) Any condition of the form "$a = v_p$"$\in$ *CONTR(Y)* if $v_p \in \{v_1,v_2,...v_k\}$ and $v_p \neq val$;

Since the rules discovered need to have minimum support, we follow the method of [AMS+95] and generate large itemsets in the first part of the algorithm. The *k-th* iteration of Apriori [AMS+95] (1) generates a set, C_k, of "candidate itemsets", whose support needs to be determined; (2) then evaluates the support of each candidate itemset from the dataset D and determines the itemsets in C_k that are large. The set of large itemsets in this iteration is L_k. [AMS+95] observes that all subsets of a large itemset are large, which is why the process of computing C_k from the set L_{k-1} can be done efficiently. The first iteration in Apriori starts with candidate itemsets of cardinality 1. The second part of the algorithm generates rules from the support values of the large itemsets. For e.g., let $I_1 = \{X,Y\}$ and $I_2=\{X\}$. From the supports of these itemsets, the confidence of the rule *if X then Y* is computed as *support(XY) / support(X).* Given these preliminaries, we describe the algorithm next.

ZoominUR algorithm is presented in Fig. 3.1. The inputs to ZoominUR are a set of beliefs, B, and the dataset D. For each belief $X \rightarrow Y$, ZoominUR finds all unexpected rules of the form $X, A \rightarrow C$, such that $C \in$ *CONTR(Y)* and the rules satisfy minimum support and confidence requirements.

For each belief $X \rightarrow Y$, ZoominUR first generates incrementally all large itemsets that may potentially generate unexpected rules. Each iteration of ZoominUR generates itemsets in the following manner. In the *k-th* iteration we generate itemsets of the form $\{X,P,C\}$ such that $C \in$ *CONTR(Y)*. Observe that to determine the confidence of the rule $X, P \rightarrow C$, the supports of both the itemsets $\{X,P,C\}$ and $\{X,P\}$ will have to be determined. Hence in the *k-th* iteration of generating large itemsets, two sets of candidate itemsets are considered for support determination:

(1) The set C_k of candidate itemsets . Each itemset in C_k (e.g. $\{X,P,C\}$) contains (i) the body $\{X\}$ of the belief, (ii) a condition that contradicts the head of belief, (i.e. any condition $C \in$ *CONTR(Y))* and (iii) k other atomic conditions (i.e. P is a conjunction of k atomic conditions).

(2) A set $C_{k'}$ of additional candidates. Each itemset in $C_{k'}$ (e.g. $\{X,P\}$) is generated from an itemset in C_k by dropping the contradictory condition, C.

```
Inputs: Beliefs Bel_Set, Dataset D, Thresholds min_support and min_conf
Outputs: For each belief, B, itemsets Items_In_UnexpRuleB
1 forall beliefs B ∈ Bel_Set {
2  C0 = { {x,body(B)} | x ∈ CONTR(head(B)) }; C0′ = {{body(B)}}; k=0
3  while (Ck != ∅ ) do {
4     forall candidates c ∈ Ck ∪ Ck′, compute support(c)
5     Lk = {x | x ∈ Ck  ∪ Ck′, support(x) ≥ min_support }
6     k++
7     Ck = generate_new_candidates(Lk-1, B)
8     Ck′ = generate_bodies(Ck , B)
9  }
10 Let X = {x | x ∈ ∪Li, x ⊇ a, a ∈ CONTR(head(B)) }
11 Items_In_UnexpRuleB = ∅
12 forall (x ∈ X) {
13    rule_conf = support(x)/support(x-a), where a ∈ CONTR(head(B))
14    if (rule_conf > min_conf) {
15       Items_In_UnexpRuleB = Items_In_UnexpRuleB ∪ {x}
16       Output Rule " x - a → a "
17    }
18 }
19 }
```

Figure 3.1 ***Algorithm ZoominUR***

We explain the steps of ZoominUR in Fig. 3.1 now. First, given belief, *B*, the set of atomic conditions that contradict the head of the belief, *CONTR(head(B)),* is computed (as described above). Then, the first candidate itemsets generated in C_0 (step 2) will each contain the body of the belief and a condition from *CONTR(head(B)).* To illustrate this, consider an example involving only binary attributes. For the belief $x=0 \rightarrow y=0$, the set *CONTR({y=0})* consists of a single condition *{y=1}.* The initial candidate sets, therefore, are $C_0 = \{\{x=0, y=1\}\}$, $C_{0'} = \{\{x=0\}\}$.

Steps (3) through (9) in Fig. 3.1 are iterative: Steps (4) and (5) determine the supports in dataset *D* for all the candidate itemsets currently being considered and selects the large itemsets in this set.

In step (7), function *generate_new_candidates*(L_{k-1}, *B*) generates the set C_k of new candidate itemsets to be considered in the next pass from the previously determined set of large itemsets, L_{k-1}, with respect to the belief B ("$x \rightarrow y$") in the following manner:

(1) Initial condition (*k=1*): In the example (binary attributes) considered above, assume that L_0 = {{x=0, y=1},{x=0}}, i.e. both initial candidates had adequate support. Further assume that "*p*" is the only other attribute (also binary) in the domain. The next set of candidates to be considered would be C_1 = { {x=0,y=1,p=0}, {x=0,y=1,p=1} }, and C_1' = { {x=0, p=0}, {x=0, p=1}}.

In general we generate C_1 from L_0 by adding conditions of the form "*attribute op value*" to each of the itemsets in L_0. To prevent generating trivially unexpected rules, we do not add any additional condition from *CONTR(head(B)).* This process adds a finite number of conditions efficiently because of the following reasons. First, the attributes are assumed to have a finite number of unique discrete values in the dataset *D*. Only conditions involving these discrete values are considered. Second, a syntactic check can ensure that zero-support itemsets are never generated. For example, {month ≥ 10} is not added to itemsets of the form {{month ≤ 3}, X}, while it is added to {{month ≤ 12}, X}.

(2) Incremental generation of C_k from L_{k-1} when $k > 1$: This function is very similar to the *apriori-gen* function described in [AMS+95]. For example, assume that for a belief, B, "$x \rightarrow y$", *c* is a condition that contradicts *y* and that L_1 = { {x, c, p}, {x, c, q}, {x, p}, {x, q} } . Similar to the *apriori-gen* function, the next set of candidate itemsets that contain *x* and *c* is C_2 ={ {x, c, p, q} } since this is the only itemset such that all its subsets of one less cardinality that contain both *x* and *c* are in L_1.

In general, an itemset X is in C_k if and only if for the belief *B, X* contains *body(B)* and a condition *A* such that $A \in$ *CONTR(head(B))* and all subsets of *X* with one less cardinality, containing *A* and *body(B)*, are in L_{k-1}.

In step (8), as described previously, we would also need the support of additional candidate itemsets in $C_{k'}$ to determine the confidence of unexpected rules that will be generated. The function *generate_bodies*(C_k,*B*) generates $C_{k'}$ by considering each itemset in C_k and dropping the condition that contradicts the head of the belief and adding the resulting itemset in $C_{k'}$.

Once all large itemsets have been generated, steps (10) to (16) of ZoominUR generate unexpected rules of the form $x, p \rightarrow a$, where $a \in$ *CONTR(head(B))*, from the supports of the large itemsets.

3.3 Algorithm ZoomoutUR

ZoomoutUR considers each unexpected rule generated by ZoominUR and tries to determine all the other more general rules that are unexpected.

```
Inputs: Beliefs Bel_Set, Dataset D, min_support, min_conf, For each belief, B, itemsets
                                  Items_In_UnexpRule_B
forall beliefs B {
new_candidates = ∅
forall (x ∈ Items_In_UnexpRule_B ) {
   Let K = {(k, k')|k ⊂ x, k ⊇ x-body(B), k' = k - a, a ∈ CONTR(head(B))}
   new_candidates = new_candidates ∪ K
}
find_support(new_candidates)
foreach (k,k') ∈ new_candidates
   consider rule: k' → k-k' with confidence = support(k)/support(k')
   if (confidence > min_conf) Output Rule " k' → k-k' "
}
}
```

Figure 3.2. ***Algorithm ZoomoutUR***

Given a belief $X \rightarrow Y$ and an unexpected rule $X, A \rightarrow B$ computed by ZoominUR, ZoomoutUR tries to find more general association rules of the form $X', A \rightarrow B$, where $X' \subset X$, and check if they satisfy minimum confidence requirements. Such rules satisfy the following properties. First, they are unexpected since the intersection[6] of this rule with the belief results in the rule $X, A \rightarrow B$, which is already known to hold. Second, these rules are more general in the sense that they have at least as much support as the rule $X, A \rightarrow B$. Third, the itemsets {X', A} and {X',A, B} are guaranteed to satisfy the minimum support requirement (though we still have to determine their exact support) since the itemsets {X,A} and {X,A,B} are already known to satisfy the minimum support requirement.

We present an outline of the ZoomoutUR algorithm in Fig. 3.2 (because of space limitation, we cannot describe it in detail and refer the reader to the technical report [PT97b]). For each belief *B* from the algorithm ZoominUR, we have the set of all large itemsets $Items\_In\_UnexpRule_B$ (step (15) in Fig. 3.1) that contain both *body(B)* and some condition *a*, such that $a \in CONTR(head(B))$. The general idea is to take each such large itemset, *I*, and find the supports for all the subsets of *I* obtained by dropping from *I* one or more attributes that belonging to *body(B).*

4. Applications

In this section we present results from applying our methods to two real datasets: consumer purchase data from a market research firm and web logfile data gathered at a major university site.

4.1 Marketing Application

We tested our algorithm on consumer purchase data from a major market research firm. We pre-processed this data by combining different data sets into *one* table describing the purchases of carbonated beverages and containing 36 discrete attributes. These attributes pertain to the characteristics of the purchasing transaction and the store and demographic data about the shopper's family[7]. Some demographic attributes include age and sex of the shopper, occupation, income and the presence of children in the family and size of the household. Some transaction-specific attributes include type of item purchased, coupon usage (whether the shopper used coupons to get a lower price), availability of coupons and presence of advertisements for the product purchased. The resulting dataset had 87437 records, each consisting of 36 *discrete* fields, the levels of which range from 2 to 12 distinct values.

We compiled 15 beliefs about the data in this domain that fall into three groups: (1) Usage of coupons, e.g. "*young shoppers with high income tend not to use coupons*". (2) Purchase of diet vs. regular drinks, e.g. *"shoppers in households with children tend to purchase regular beverages more than diet"*. (3) Day of shopping, e.g. *"professionals tend to shop more on weekends than on weekdays"*. Some of these beliefs were from experts and others were learned from data and subsequently selected by the expert as "beliefs". In this marketing example, all beliefs were expressed as association rules, and ZoomUR, therefore, generated only associations.

We generated on average 40 rules per belief (a total of about 600 rules), many of which were interesting, not just by definition of unexpectedness, but to experts as well. Being able to discover some rules really interesting to experts with more ease than having to look through thousands of rules [BMU+97] illustrates the advantage of our simple, yet effective approach. Some representative examples are:

Belief: *Shoppers with children tend to buy regular rather than diet beverages* (presumably because children prefer regular to diet beverages). While, this holds in general in

[6] As described in part (b) of the definition of unexpectedness in Section 2.

[7] We note that this is unnormalized data containing in one file both transaction and demographic data.

the data, ZoominUR discovered the unexpected rule:

- *When there is a large store advertisement, shoppers with children buy diet beverages.*

This is a really interesting rule to an expert, because it indicates that under a certain condition (the presence of a large advertisement in the store), a population that usually bought products of one kind, buy exactly the *opposite* product. If these advertisements represent a sale in diet beverages, this rule provides evidence of the success of the advertising campaign.

Belief: *Professionals tend to shop more on weekends than on weekdays* (presumably because they are busier during the week). It turns out that this belief by itself is "true" (holds with high confidence in the data). However, ZoominUR discovered some very interesting rules such as:

- *In December, professionals tend to shop more on weekdays than on weekends*.
- *Professionals in large households tend to shop more on weekdays than on weekends*.

Post-discovery, these rules seem to make sense, perhaps because the holiday season in December makes professionals shop more often on weekdays and because large households may have shopping demands far more often than smaller households, which could make professionals shop more often. For this belief, ZoomoutUR also discovered that:

- *In December, shoppers in general shop more on weekdays than on weekends.*

This gives some evidence that it may not necessarily be a "professionals in december" effect, but shoppers in general in December shop more on weekdays. Also observe that this rule is *not* just a refinement of the belief, but a much different rule (although still unexpected).

Belief: *Retired shoppers tend to use coupons for their purchases* (because they can shop with more freedom and when coupons are available). For this belief, there was a direct contradiction.

Since ZoomUR in this case generates association rules, we also ran Apriori algorithm on this dataset[8] and generated over 40,000 rules, many of which were irrelevant or obvious. However, this is not surprising since the objective of Apriori is to generate *all strong* association rules. Our experiments demonstrate that the generation of these irrelevant or obvious rules can be avoided to a large extent by using prior domain knowledge (expressed as beliefs) to seed the search process.

4.2 Mining Web Logfile Data

We also tested our method on Web logfile data tracked at a major university site. The data was collected over a period of 8 months from May through December 1997 and consisted of over 280,000 hits. Some of the interesting rules in this application involve comparison operators. For example, temporal patterns holding during certain time intervals need to be expressed with conditions of the form "$20 \leq week \leq 26$" (Sep. 10 through Oct. 29 in our example). We generated 11 beliefs about the access patterns to pages at the site. An example of a belief is:

Belief: *For all files, for all weeks, the number of hits to a file each week is approximately equal to the file's average weekly hits.*

Note that this belief involves aggregation of the Web logfile data. To deal with this, we created a user-defined *view* on the Web logfile and introduced the following attributes: *file*, *week_number*, *file_access_cnt*, *avg_access_cnt_file*, *stable_week*. The *file_access_cnt* is the number of accesses to *file* in the week *week_number*. The *avg_access_cnt_file* is the average weekly access for *file* in the dataset. The *stable_week* attribute is 1 if *file_access_cnt* lies within two standard deviations around *avg_access_cnt_file* and is 2(3) if *file_access_cnt* is higher (lower) . The above belief can then be expressed as $True \rightarrow stable\_week=1$. Though this belief was true in general (holds with 94% conf. on the view generated), ZoominUR discovered the following unexpected rules:

- *For a certain "Call for Papers" file, in the weeks from September 10 through October 29, the weekly access count is much higher than the average*. i.e.

$file = cfp\_file,\ week\_number \geq 20,\ week\_number \leq 26 \rightarrow stable\_week=2.$

What was interesting about this rule was that it turned out to be a Call-for-papers for the *previous* year and the editor of the Journal could not understand this unusually high activity! The file was removed from the server.

- *For a certain job opening file, the weeks closest to the deadline had unusually high activity.*

$file = job\_file,\ week\_number \geq 25,\ week\_number \leq 30 \rightarrow stable\_week=2.$

This pattern is not only unexpected (relative to our belief) but is also actionable because the administrators can expect a large number of applications and should prepare themselves for this. Also, this pattern can prompt the administrators to examine IP domains that do not appear in the Web log accesses and target them in some manner.

We would like to make the following observations based on our experiments with the Web application. First, as the examples show, we need to incorporate *comparisons* since many of the interesting patterns *are* expressed in these terms. Second, the raw web access log data has very few fields, such as *IP_Address*, *File_Accessed*, and *Time_of_Access*. Without beliefs it would be extremely difficult to discover relevant patterns from this "raw" data.

[8] In the process we extended Apriori to handle discrete rather than binary attributes.

Beliefs provide valuable domain knowledge that results in the creation of several user-defined views and also drive the discovery process.

5. Conclusions

In this paper, we presented an algorithm for the discovery of unexpected patterns based on our definition of unexpectedness proposed in [PT97a]. This algorithm uses a set of user-defined beliefs to seed the search for the patterns that are unexpected relative to these beliefs. We tested our algorithm on two "real-world" data sets and discovered many interesting patterns in both data sets.

These experiments demonstrated two things. First, user-defined beliefs can drastically reduce the number of irrelevant and obvious patterns found during the discovery process and help focus on the discovery of unexpected patterns. Second, user-defined beliefs are crucial for the discovery process in some applications, such as Weblog applications. In these applications, important patterns are often expressed in terms of the user-defined vocabulary [DT93] and beliefs provide the means for identifying this vocabulary and driving the discovery processes.

As explained in the introduction, we do not describe how to generate an initial system of beliefs. To generate such beliefs, we use the methods described in [ST96b]. However, more work needs to be done to extend the belief generation methods considered in [ST96b]. In the future, we plan to work on such extensions. We are also working on incorporating predicates and aggregations into the beliefs and on using them in the discovery processes.

References

[AIS93] Agrawal, R., Imielinski, T. and Swami, A., 1993. Mining Association Rules Between Sets of Items in Large Databases. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pp. 207-216.

[AMS+95] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H. and Verkamo,A.I., 1995. Fast Discovery of Association Rules. In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. eds., *Advances in Knowledge Discovery and Data Mining*. AAAI Press.

[AT97] Adomavicius, G., and Tuzhilin, A., 1997. Discovery of Actionable Patterns in Databases: The Action Hierarchy Approach. In *Proc. of the Third Intl. Conference on Knowledge Discovery and Data Mining (KDD 97).*

[BMU+97] Brin, S., Motwani, R., Ullman, J.D., and Tsur, S., 1997. Dynamic Itemset Counting and Implication Rules for Market Basket Data. *Procs. ACM SIGMOD Int. Conf. on Mgmt. of Data, pp.255-264.*

[DT93] Dhar, V., and Tuzhilin, A., 1993. Abstract-Driven Pattern Discovery in Databases. *IEEE Transactions on Knowledge and Data Engineering*, December 1993.

[F97] Forbes Magazine, Sep. 8, 1997. *Believe in yourself, believe in the merchandise*, pp.118-124.

[FPM91] Frawley, W.J., Piatetsky-Shapiro, G. and Matheus, C.J., 1991. Knowledge Discovery in Databases: An Overview. In Piatetsky-Shapiro, G. and Frawley, W.J. eds., *Know. Disc. in Databases*. AAAI/MIT Press, 1991.

[FPS96] Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., 1996. From Data Mining to Knowledge Discovery: An Overview. In Fayyad, U.M.,Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. eds., *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press.

[KMR+94] Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H. and Verkamo, A.I., 1994. Finding Interesting Rules from Large Sets of Discovered Association Rules. In *Proc. of the Third International Conference on Information and Knowledge Management*, pp. 401-407.

[LH96] Liu, B. and Hsu, W., 1996. Post-Analysis of Learned Rules. In *Proc. of the Thirteenth National Conf. on Artificial Intelligence (AAAI '96)*, pp. 828-834.

[LHC97] Liu, B., Hsu, W. and Chen, S, 1997. Using General Impressions to Analyze Discovered Classification Rules. In *Proc. of the Third Intl. Conf. on Knowledge Discovery and Data Mining (KDD 97),* pp. 31-36.

[PS91] Piatetsky-Shapiro,G.,1991. Discovery,Analysis and Presentation of Strong Rules. In Piatetsky-Shapiro and Frawley, eds.,*Know. Disc. in Databases*.AAAI/MIT Press.

[PSM94] Piatetsky-Shapiro, G. and Matheus, C.J., 1994. The Interestingness of Deviations. In *Proc. of AAAI-94 Workshop on Know. Discovery in Databases*, pp. 25-36.

[PT97a] Padmanabhan, B. and Tuzhilin, A., 1997. On the Discovery of Unexpected Rules in Data Mining Applications. In *Procs. of the Workshop on Information Technology and Systems (WITS '97),* pp. 81-90.

[PT97b] Padmanabhan, B. and Tuzhilin, A., 1997. Unexpectedness as a Measure of Interestingness in Knowledge Discovery. *Working Paper #IS-97-6, Dept. of Information Systems, Stern School of Business, NYU.*

[S97] Stedman, C., 1997. Data Mining for Fool's Gold. *Computerworld*, Vol. 31,No. 48, Dec. 1997.

[ST95] Silberschatz, A. and Tuzhilin, A., 1995. On Subjective Measures of Interestingness in Knowledge Discovery. In *Proc. of the First International Conference on Knowledge Discovery and Data Mining*, pp. 275-281.

[ST96a] Silberschatz, A. and Tuzhilin, A., 1996. What Makes Patterns Interesting in Knowledge Discovery Systems. *IEEE Trans. on Know. and Data Engineering*. Spec. Issue on Data Mining, v.5, no.6, pp. 970-974.

[ST96b] Silberschatz, A. and Tuzhilin, A., 1996. A Belief-Driven Discovery Framework Based on Data Monitoring and Triggering. *Working Paper #IS-96-26, Dept. of Information Systems, Stern School of Business, NYU.*

[SVA97] Srikant, R., Vu, Q. and Agrawal, R. Mining Association Rules with Item Constraints. In *Proc. of the Third International Conference on Knowledge Discovery and Data Mining (KDD 97),* pp. 67-73.